Deploying Stateful Applications with Cisco HyperFlex CSI

Prerequisites for Deploying Stateful Applications with Cisco HyperFlex CSI

The following prerequisites must be met prior to deploying stateful applications using the HyperFlex CSI storage integration.

- Cisco HyperFlex cluster is installed and running 4.0(1a) or later.
- Kubernetes support must be enabled in HyperFlex Connect.
- The Cisco HyperFlex CSI integration has been deployed.

Administrator Host

In this chapter, the Administrator Host is simply a linux-based system that is used to administer run kubectl commands, etc. against the Kubernetes cluster. While this is typically a separate system (VM) that is not part of the Kubernetes cluster, you can use one of the Kubernetes nodes as the administrator host if you do not wish to install/manage a separate system (VM).

Deploying Stateful Applications

To deploy stateful applications, perform the following procedures:

Creating a Persistent Volume Claim

A Persistent Volume Claim is a simply a request for storage by a user. Users specify their storage requirements, the size or capacity of the storage required, and other options. Depending on the associated Storage Class, the storage requirements are routed to the appropriate provisioner which knows how to provision the requested storage, and make it available to Kubernetes

Procedure

Step 1 On the administrator host, create a file named “message-board-pvc.yaml” with the following contents

Example:

```
administrator-host:hxcsi$ cat ./message-board-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: message-board-pvc
spec:
  storageClassName: csi-hxcsi-default
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Step 2 On the administrator host, use the `kubectl create -f` command to create the Persistent Volume Claim.

Example:

```
administrator-host:hxcsi$ kubectl create -f ./message-board-pvc.yaml

persistentvolumeclaim/message-board-pvc created
```

Step 3 n the administrator host, use the `kubectl get pvc` command to verify the Persistent Volume Claim was created and is successfully bound to a Persistent Volume.

Example:

```
administrator-host:hxcsi$ kubectl get pvc
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS        AGE
message-board-pvc   Bound    pvc-8069462e-662c-11e9-a163-005056a086d9   10Gi       RWO
        csi-hxcsi-default   105s
```

Deploy Stateful Kubernetes Workload

Kubernetes workloads come in various forms, such as Pods and Deployments regardless of the type of Kubernetes workload, each can leverage persistent storage using the Cisco HyperFlex CSI integration and Persistent Volume Claims. The following shows the deployment of a sample open source application called Cisco Message Board that can be used to test the Cisco HyperFlex CSI integration. You can also test with your own applications following the same methodology and procedures.

Procedure

Step 1 On the administrator host, create the YAML file which defines the workload to be deployed.

Example:

The following shows the YAML file for the example Cisco Message Board application which will create both a Kubernetes Deployment and a Kubernetes Service which will allow for connecting to the deployed Cisco Message Board application through a NodePort.

Note That we are referencing the Persistent Volume Claim name in the "volumes" section of the Kubernetes Deployment definition. In this example, the Persistent Volume bound to the "message-board-pvc" Persistent Volume Claim will be mounted inside the "message_board:version1" container at the "/sqldb" location (path)

```
administrator-host:hxcsi$ cat ./message-board-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: message-board
    labels:
        app: message-board
spec:
    replicas: 1
    selector:
        matchLabels:
            app: message-board
    template:
        metadata:
            labels:
                app: message-board
                name: message-board
        spec:
            volumes:
                - name: demovolume1
                  persistentVolumeClaim:
                    claimName: message-board-pvc
            containers:
            - name: message-board
              image: michzimm/message_board:version1
              ports:
              - containerPort: 5000
              volumeMounts:
                  - mountPath: "/sqldb"
                    name: demovolume1
---
apiVersion: v1
kind: Service
metadata:
  name: message-board
  labels:
    name: message-board
  namespace: default
spec:
  type: NodePort
  ports:
  - port: 5000
    nodePort: 30002
  selector:
    name: message-board
```

Step 2 On the administrator host, use the `kubectl create -f` command to create the Deployment and Service.

Example:

```
administrator-host:hxcsi$ kubectl create -f ./message-board-deployment.yaml
deployment.apps/message-board created
service/message-board created
```

Step 3 On the administrator host, use the `kubectl get pods` command to check the status of the deployed Pods.

Example:

```
administrator-host:hxcsi$ kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
csi-attacher-hxcsi-0             2/2     Running   0          3h51m
csi-nodeplugin-hxcsi-9fgsf       2/2     Running   0          3h51m
csi-nodeplugin-hxcsi-qqvwj       2/2     Running   0          3h51m
csi-provisioner-hxcsi-0          2/2     Running   0          3h51m
message-board-6df65d6b59-49xhq   1/1     Running   0          95s
```

Step 4 On the administrator host, use the `kubectl get services` command to check the status of the deployed Service.

Example:

```
root@administrator-host:hxcsi$ kubectl get services
NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
csi-attacher-hxcsi      ClusterIP   10.98.79.159     <none>        12346/TCP        3h53m
csi-provisioner-hxcsi   ClusterIP   10.99.73.185     <none>        12345/TCP        3h53m
kubernetes              ClusterIP   10.96.0.1        <none>        443/TCP          4h24m
message-board           NodePort    10.107.227.152   <none>        5000:30002/TCP   2m59s
```

For the sample Cisco Message Board application, the service is configured using “NodePort” and port “30002” meaning the application should be a up and running and accessible by pointing your web browser to any Kubernetes node IP address and port “30002”. For example: http://<k8s-worker1>:30002